UNIVERSITY OF CALIFORNIA, DAVIS
Department of Electrical and Computer Engineering

EEC 172 Spring 2021

LAB 1 (100 points): Development Tools Tutorial and Lab Exercise

Lab Due: Apr. 11

Objective: This tutorial will cover the basic software development tools that will be used in this course, including Code Composer Studio (CCS), CCS UniFlash, and the TI Pin Mux Tool.

Equipment Needed

- CC3200 LaunchPad (CC3200-LAUNCHXL)
- USB Micro-B plug to USB-A plug cable (included in Kit)
- Optional: USB Flash Drive for saving code

Note: Do not lose the shorting blocks (Jumpers) that come with the board. They are used to configure different hardware functions on the board and may be needed for this and future labs.

Development Tool Installation Procedure

The following software must be installed on your own computer to use it for project development. Note that to download some of the TI software, you will need to set up a TI account. Establish your own account to download the software instead of trying to get it in some other way.

1. **Code Composer Studio (CSS) Integrated Development Environment (v9.3.0)**
   http://processors.wiki.ti.com/index.php/Download_CCS
   Download and install the latest version of CCS. You will have access to the full-featured version (not code-size or time limited) when using the CC3200 Launchpad.

2. **CC3200 Software Development Kit (v1.4.0)** and **CC3200SDK-ServicePack (v1.0.1.13-2.11.0.1)**
   http://www.ti.com/tool/cc3200sdk
   You will be asked to create a TI account.

3. **TI Pin Mux Tool (v4.0.1223)**
   http://processors.wiki.ti.com/index.php/TI_PinMux_Tool
   You can use either the Cloud version of the Pin Mux Tool or the stand-alone desktop version.

4. **CCS UniFlash (3.4.1.00012)**
   http://processors.wiki.ti.com/index.php/CCS_UniFlash_v3.4.1_Release_Notes

5. **PuTTY** or **Tera Term**
   You will use a terminal emulator program for serial input and output to you CC3200. You can use any terminal emulator program that you prefer. The lab machines will have PuTTY (available at http://www.putty.org/) and Tera Term (available from https://ttssh2.osdn.jp/index.html.en).

Part I. Testing Example Programs in Code Composer Studio

In this part, you will create a workspace and import two example programs to test with your CC3200 processor.

A. Blinky example program

1. Launch Code Composer Studio (CCS) from the desktop icon or from the Start Menu.

   CCS will prompt you to select a directory for your workspace. Enter a directory in your user space, such as:

   C:\Users*your_name*\workspace_v9_cc3200

   Do ***NOT*** check the box "Use this as the default and do not ask again" on any of the lab machines since these machines will be shared among many users. Even on your own machine, we recommend that you avoid this option.

2. Select **Import Project** from the Getting Started page or select **Import CCS Projects…** from the Project pull-down menu in the Edit Perspective of CCS.

   Browse to **C:\ti\CC3200SDK_1.4.0\cc3200-sdk\example\blinky**

   Check the box to **Automatically import referenced projects found in the same search-directory** (if the checkbox is not disabled) and click **Finish**.

3. In Project Explorer, right-click on **blinky** and select **Properties**. Change the Compiler version from to TI v18.12.04.LTS or higher to get rid of a warning message. You should explore other Properties. For example, expand ARM Linker and select Basic Options in order to see the default heap and stack sizes for this project. When you are done examining project properties, click **OK.**

4. Also, in the Properties window, go to the **Processor Options** page and from the drop-down menu in front of the Specify floating point support (--float_support), select the **FPv4SPD16** option.

5. Click the hammer icon or select **Project > Build Project** to build the blinky project. The project should build without any errors or warnings.

6. Connect the CC3200 LaunchPad to your host computer using the supplied USB cable.

7. Click the bug icon or select **Run > Debug** to launch the Debug Perspective. In the older versions of Code Composer Studio, the first time you attempt to run a program on the CC3200, you might get the following message:

   You require a target configuration to start a debug session. Do you want to create a new target configuration file and open it in the editor?

   If so, Click **Yes**.

   Give the new Target Configuration file a name, such as **cc3200.ccxml**.

   Click the box "**Use shared location**" so that this configuration file can be used for all of your projects in this workspace. Then click **Finish**.

For the Connection, select **Stellaris In-Circuit Debug Interface** and select the box for "**CC3200**" for the board or device. Then scroll to the right of the window and select **Save**. You should now see CC3200.ccxml listed under the User Defined folder in the Target Configurations window. Make sure that CC3200.ccxml is selected as the **default** target configuration. Close the cc3200.ccxml window and click the bug icon or **Run > Debug** again. This time, your program will be loaded into the CC3200's RAM. (Make sure that the CC3200 LaunchPad is still connected to the PC via the USB cable.) If you don't see this message you can ignore the step 7.

8. Now the Micro Controller is programmed and has paused at the main function. Click on the green "Play" button icon or select **Run > Resume** or type **F8** to run the program. You should see the red, yellow and green LEDs blink in a continuous sequence. Examine the program to see how it works.

9. Make a simple modification to make the LEDs blink faster than in the default program. For example, you can easily make the LEDs blink two times to ten times faster with simple code modifications. Download the code again and verify that the program works as expected.

B. Uart_demo example program

Follow the steps above to import the **uart_demo** project. You can make the project Active for Debugging by clicking on the project's name in the Project Explorer window inside the Code Composer Studio App. Build and download the program as you did for the blinky program. Before running the program, you will need to open a terminal window using PuTTY or TeraTerm (Although not recommended, you can open the Terminal window from the CCStudio as well).

Open the Windows **Device Manager** (Start Menu -> right click on Computer -> select Manage -> select Device Manager from under System Tools). Expand the Ports (COM & LPT) entry and determine which COM port is used for communicating with the CC3200. You will need to open this COM port in PuTTY or TeraTerm.

Open the terminal emulator program and configure it for a serial terminal connection to the COM port used by the CC3200. Set the baud rate to 115200.

Once your terminal window is open, run the uart_demo program in the CCS Debug window. Verify that the program echoes text that you type into the terminal window using the keyboard.

Remember that while flashing the CC3200, the serial terminal connection should be closed and reopened later.

Part II. Lab1 Application Program Exercise

In this part, you will develop a simple program that interfaces a console window to switches and LEDs on the CC3200 LaunchPad.

1. **Create a project in CCS**

   Instead of creating a new, empty project, it is easier to import an example project from the CC3200SDK and modify that project according to the new project specs. In this case, we recommend that you start with the ***blinky*** project since it is similar to the new project you will develop. Note that you cannot import ***blinky*** into your project when it is already there. However, you can rename the ***blinky*** project to a new name, such as ***Lab1***. That way, you could re-import the original ***blinky*** project into your workspace, if desired.

2. **Configure the pins for your project using the TI Pin Mux Tool**

   The TI Pin Mux Tool is a utility used to select the appropriate pin multiplexing configuration to satisfy the application requirements. This tool makes it easy to understand the various pin configuration options and to implement your desired pin configuration without error. You can use either the Cloud-hosted Pin Mux Utility or the stand-alone desktop version. In this section, we will describe using the desktop version. For information on using the TI Cloud-hosted tool, see the following video:

   https://www.youtube.com/watch?v=Q8yby_i3N_M

   - Launch the TI Pin Mux Tool from either the Start menu, the desktop icon, or from the cloud version at dev.ti.com, if it exists. Select **CC3200** as the device and click on the **Start** button.

   - We will first configure the desired GPIO signals. For this project, we want to interface to the following devices/pins on the CC3200 LaunchPad: Red, Yellow and Green LEDs, SW2 and SW3, P18 on the P2 header, and the UART Rx and Tx pins which interface to the console window. To determine which pins or GPIO signals you need to specify, consult the LaunchPad schematic (CC3200-LAUNCHXL_SCH_Rev4p1-a.pdf) available on the course website. Make sure that you can verify the following signal names and pin numbers on the schematic. (It will be very important that you consult this schematic when deciding which pins to use for hardware interfacing. Many of the pins that are brought to the headers are already being used on the LaunchPad board and are not actually available for general use.)

Signal Name	Pin Number	Device
GPIO_9	64	Red LED
GPIO_10	1	Yellow LED
GPIO_11	2	Green LED
GPIO_13	4	SW3
GPIO_22	15	SW2
GPIO_28	18	P18 on P2 header

   - Select the GPIO peripheral in the Peripherals tab on the left.

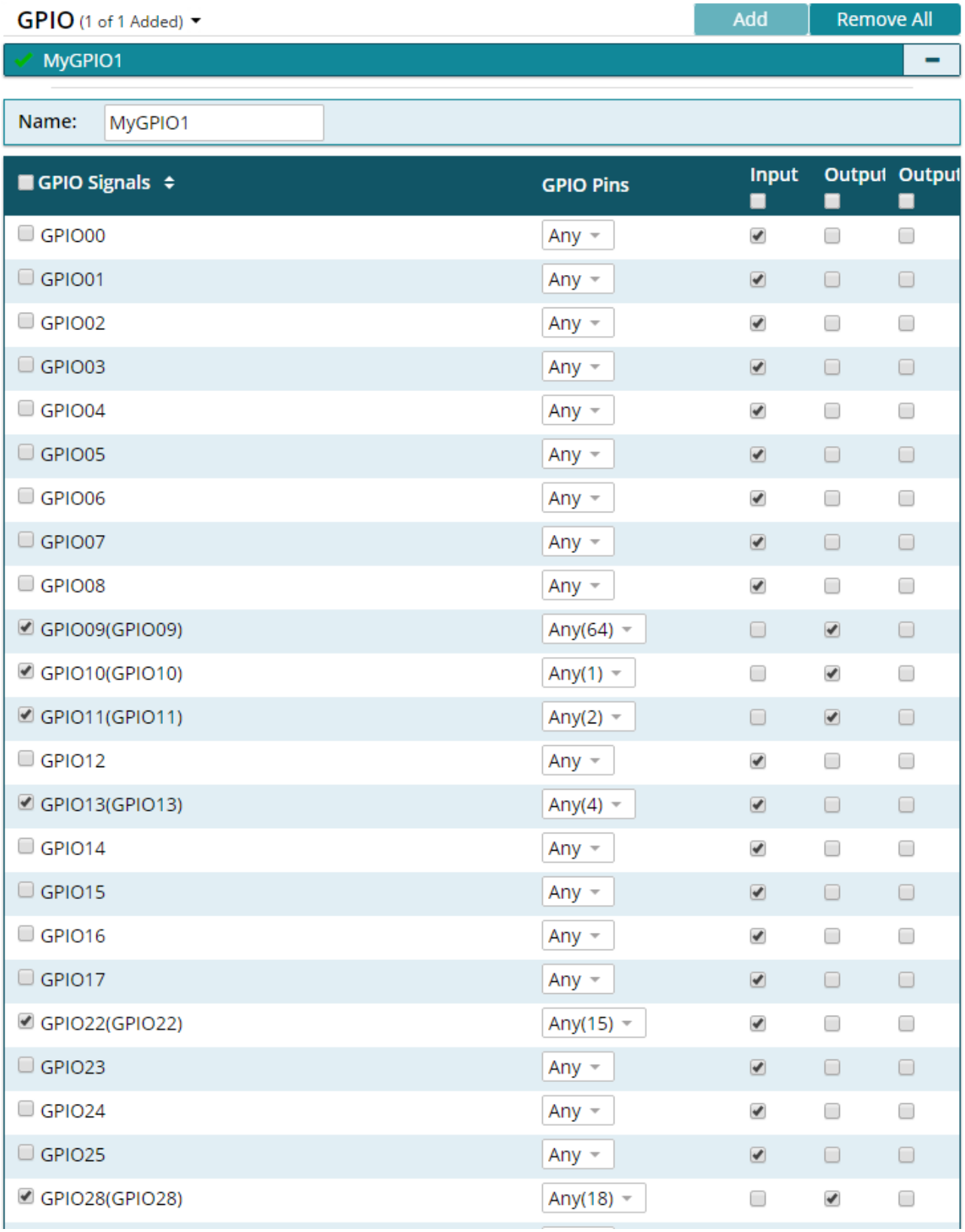
   - Click on the Add button in the Requirements tab so that you can add GPIO signals.

   - Unselect all the GPIO signals by unchecking the box in the top left labeled GPIO Signals. Next,

add the GPIO signals that you want to use as GPIO in your design.

- Check the appropriate box for Input, Output or Output OD (Open Drain) for each of your signals. In this design, the switches should drive input pins and outputs signals (*not* open drain outputs) will drive the LEDs and P18.

- Verify that the pin numbers match the ones in the table above. If they don't match, use the pull-down option to select the correct pin number. Your GPIO section should look like the following screenshot. (The name of the module is arbitrary.)

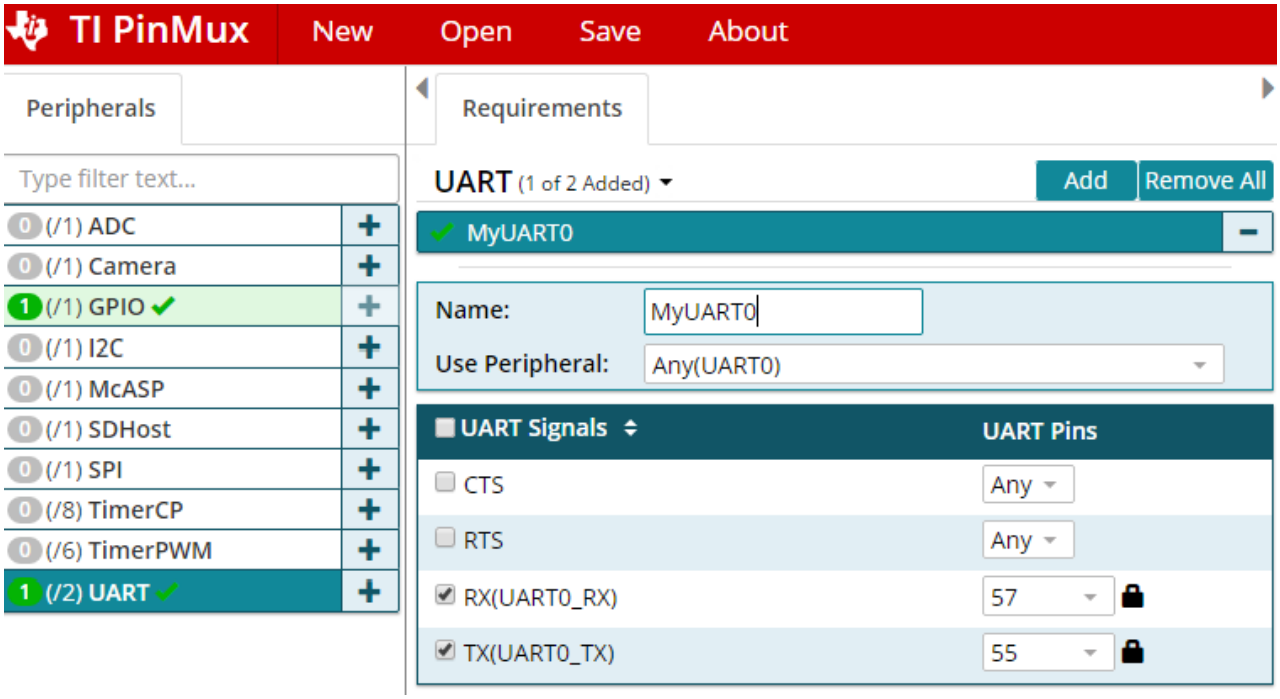


PinMux Configuration of GPIO signals

– Next configure the UART peripheral. Note that you will need to configure the UART0_RX to pin number 57 and UART0_TX to pin number 55 as these are not the default pin assignments for these signals on the CC3200 LaunchPad. The lock icon shows that you have locked that pin number

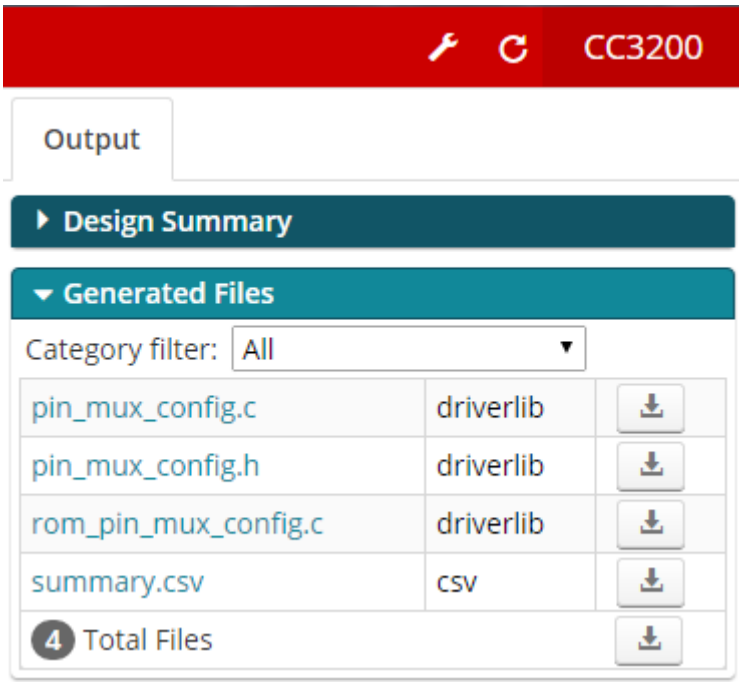
to the specified signal. For details, see the UART MUXING and the EMULATION sub- circuits on the CC3200 LaunchPad schematic. Your UART configuration should look similar to the screenshot below.

Signal Name	Pin Number	Device
UART0_RX	57	FTDI_TX
UART0_TX	55	FTDI_RX



PinMux Configuration of UART signals

– Save your design to your project directory. For example, you can use the name lab1 and the file will be stored as lab1.pinmux. You can open this file with the Pin Mux Tool to change your pin configuration instead of starting over.



PinMux output files

- Select the download button to download pin_mux_config.c and pin_mux_config.h to your project directory. The download button is the down-arrow icon to the right of the filename, as shown in the screenshot below. Open these files and verify the code that has been generated to configure the pins. (You could use the rom_pin_mux_config.c file instead of the pin_mux_config.c file if you choose.) Note that these files will replace the pinmux.c and pinmux.h used in the original blinky project.

- Remove the old pinmux.c and pinmux.h files from your project directory. In main.c, modify the #include to use "pin_mux_config.h" instead of "pinmux.h".

3. <u>Modify your program code to meet the application specifications</u>

The program specifications are as follows:

- When your program starts, it should display a message on the console window (i.e. TeraTerm or other terminal emulator) with a header and usage instructions as shown below. A good example project to look at for basic UART functions is uart_demo in the CC3200SDK examples folder.

```
*************************************************
     CC3200 GPIO Application
*************************************************


*************************************************
     Push SW3 to start LED binary counting
     Push SW2 to blink LEDs on and off
*************************************************
```

- Your program should poll the SW3 and SW2 switches on the CC3200 LaunchPad. When SW3 is pressed, you should start a binary counting sequence on the LEDs, counting from 000 – 111 *continuously* on the three LEDs. The count should be relatively slow so that you can see each count value easily. You should also print a message to the console "SW3 pressed". This message should not be printed again until after SW2 has been pressed.

  When SW2 is pressed, your program should blink the LEDs ON and OFF in unison. Again, the blink pattern should be relatively slow so that you can see each pattern clearly and easily. You should print the message "SW2 pressed" to the console. This message should not be printed again until after SW3 has been pressed.
- Set the output signal P18 high whenever SW2 is pressed and low whenever SW3 is pressed. You can verify this signal using an oscilloscope.

  NOTE: For polling the switches and controlling the outputs, you can use CC3200 Peripheral Driver Library APIs such as GPIOPinRead() and GPIOPinWrite(). The CC3200 Peripheral Driver Library User's Guide can be found at

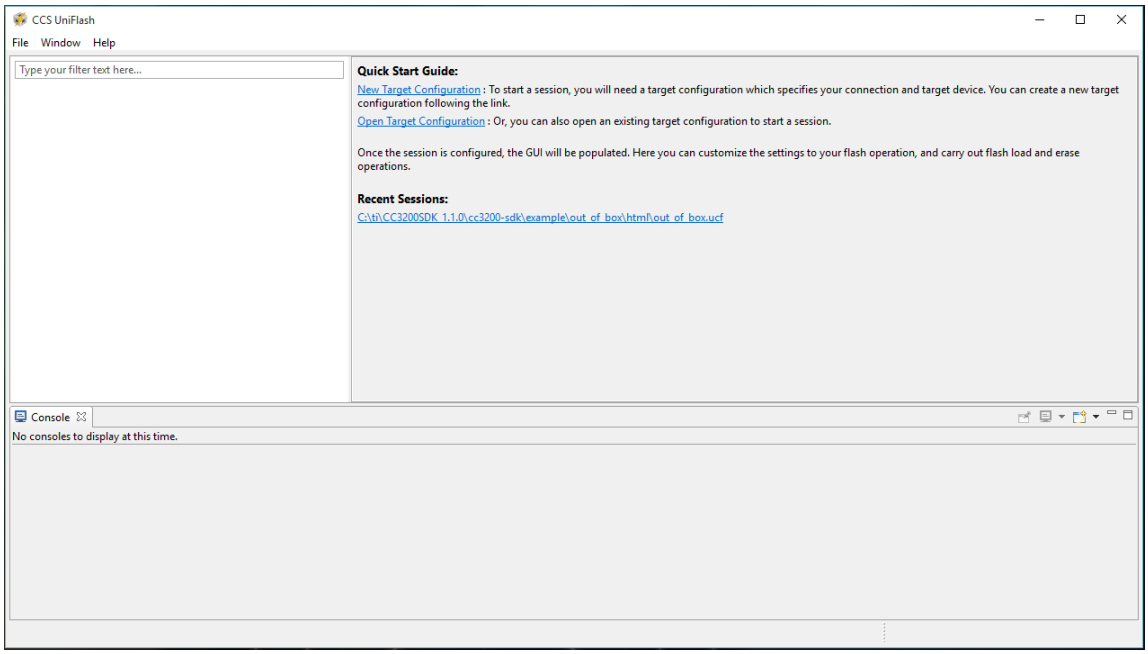
  http://software-dl.ti.com/ecs/cc31xx/APIs/public/cc32xx_peripherals/latest/html/index.html

  Verify your program using the CCS Debug mode.

Part III. Using CCS UniFlash

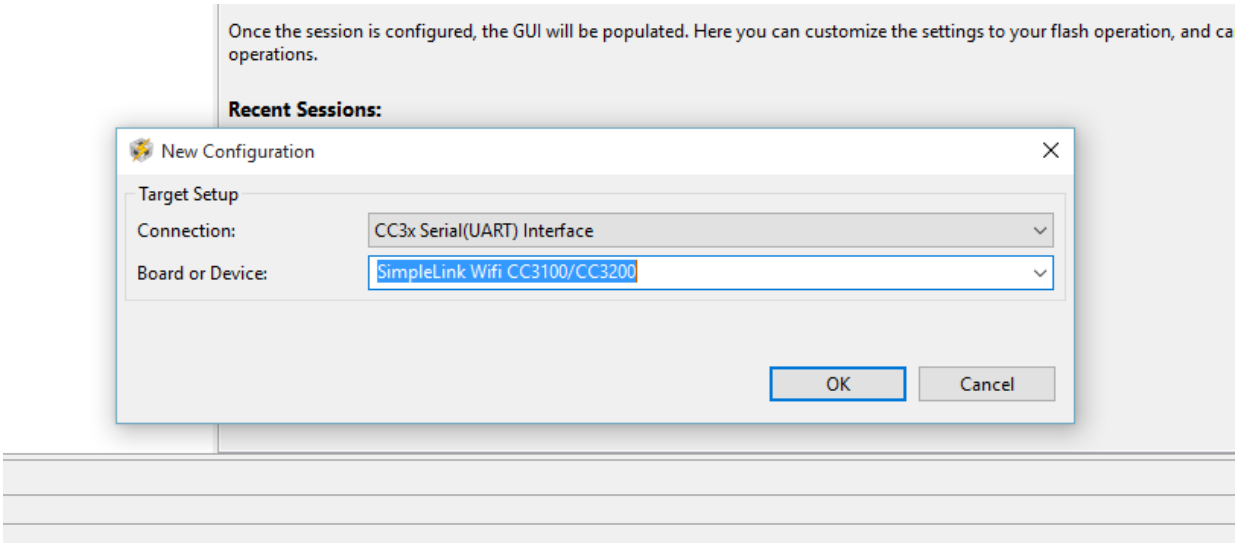
In this part, you will learn how to program the external flash memory so that the program will remain in non-volatile memory through power cycles. This section will demonstrate how to load the ***blinky*** program onto the external flash device on the CC3200 LaunchPad.

On launching the UniFlash tool, you will see a window as shown below. To start, click on the *New Target Configuration* link under the Quick Start Guide heading, or under the File > New Configuration on the menu bar. This should open up the new configuration window.



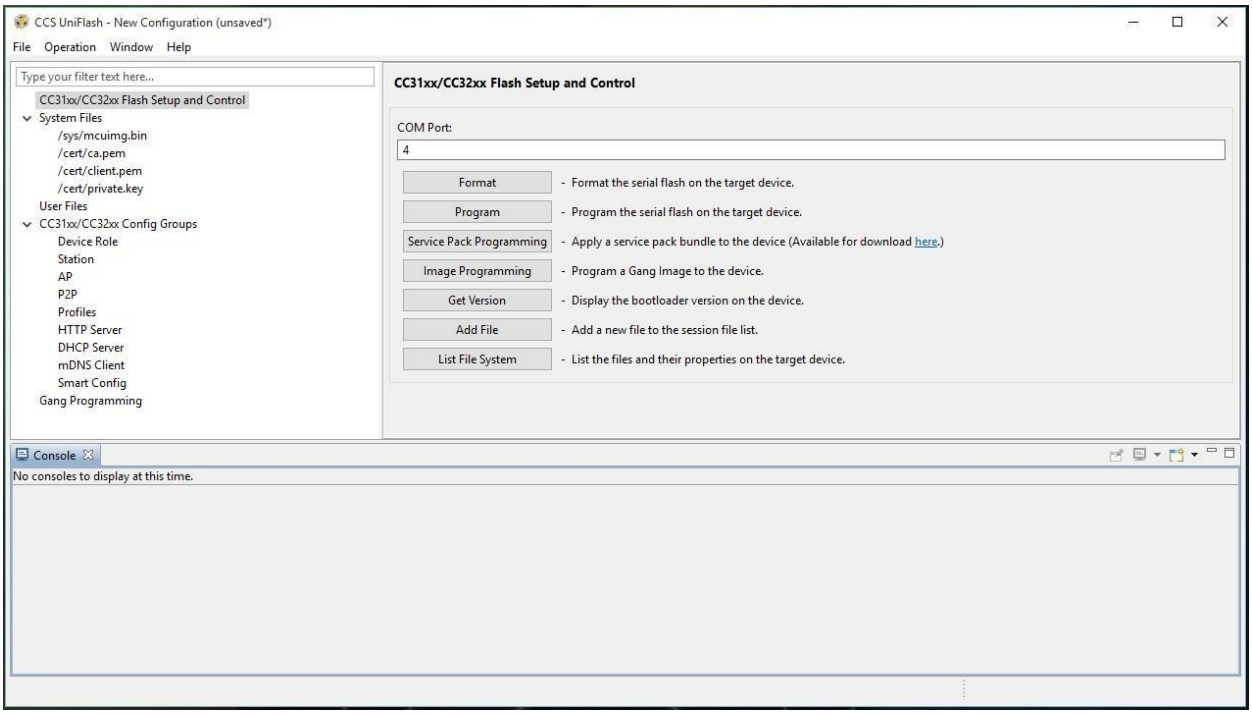
Launch Screen of the CCS UniFlash tool

From the Connection drop down menu, select the '**CC3x Serial(UART) Interface**' option, and the '**SimpleLink Wifi CC3100/CC3200**' for the Board or Device in the screen-shot below. After the selections are made, click **OK**.



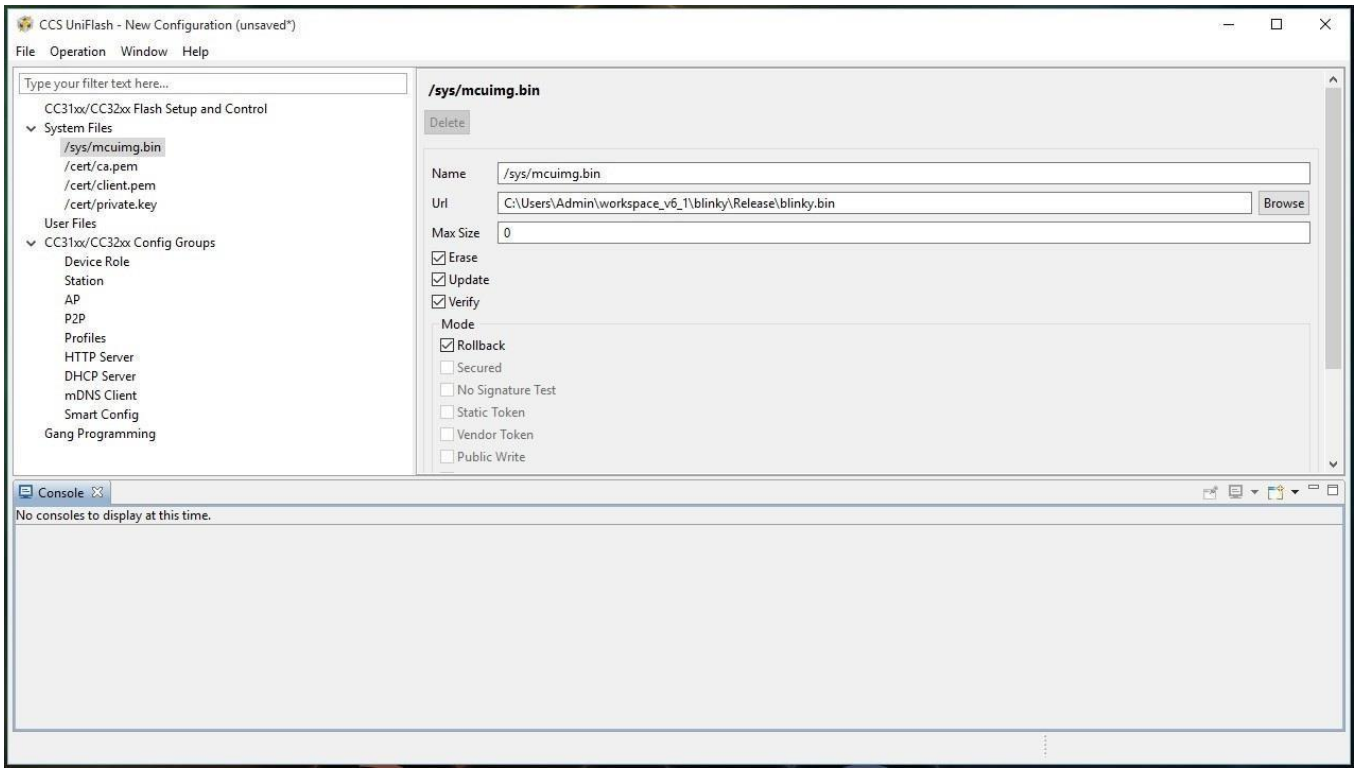
New Configuration Window

This will open up the main screen shown below. First, the correct COM Port for the device must be identified to the UniFlash tool. If you are unsure of which COM Port to use, plug in the CC3200 Launchpad and open up the Device Manager to check. In this example, the CC3200 Launcpad was on COM Port 4. Enter the COM Port number into the text box, and then select the '/sys/mcuimg.bin' option on System Files on the left of the screen.



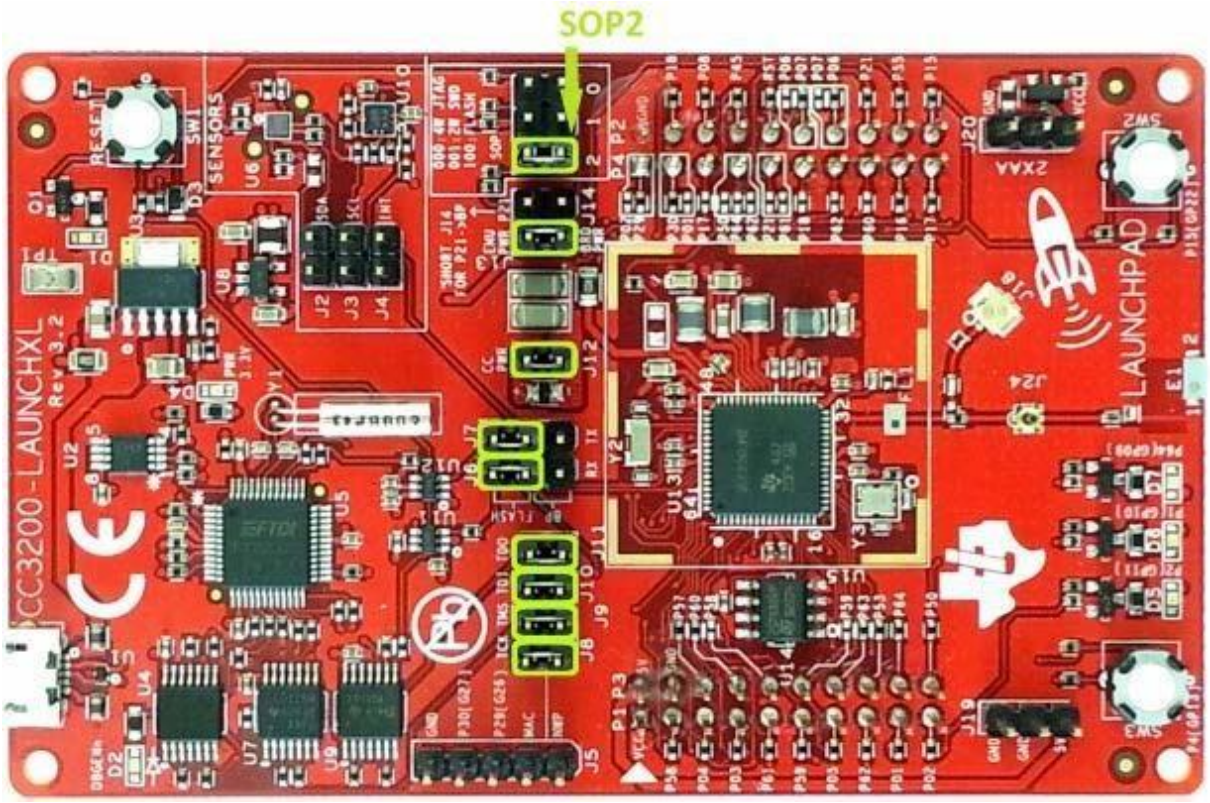
CCS UniFlash Flash Setup and Control Screen

This panel will allow you to identify the compiled .bin file that will be loaded into the flash memory. Click on the *Browse* button next to the Url text box, and locate the .bin file that you would like to load into flash. For this example, the compiled Blinky example program, blinky.bin was selected from the CCS workspace. In addition, make sure the *Update*, and *Verify* check boxes are selected. The UniFlash tool should now be ready to flash the CC3200 LaunchPad hardware.



System File Options Screen

Before connecting the CC3200 Launchpad to the computer, a jumper will need to be placed on SOP2 as shown below. This will allow the flash memory to be programmed by CCS UniFlash, and will prevent the previously flashed program from being executed.



Minimum Jumper Configuration for Flashing the CC3200

(Source: Figure 1 of the CC3200 SimpleLink Wi-Fi and IoT Solution With MCU LaunchPad Getting Started Guide)

Once you have verified the jumpers are correctly set, connect the Launchpad to the computer, and return to the *Flash Setup and Control* panel that was shown earlier and click the **Program** button. This should successfully program the board. To see the program run, disconnect the Launchpad from the computer, and remove the SOP2 jumper, and reconnect the board to the computer. (Note: Do not lose the jumper since you will often need to use it.) The blinky.bin file should now run at launch instead of the out of the box demo program (or whatever program was previously programmed in flash). If this does not work, double check that the necessary jumpers are placed on the Launchpad, as well as reviewing the options on the *'/sys/mcuimg.bin'* panel are set correctly.

Use the CCS UniFlash utility to program your application into the on-board serial Flash chip Once your application program from Part II works, load it into the serial Flash chip on the CC3200 LaunchPad so that your program can run without downloading the code from CCS into RAM in Debug Mode.

To test your program, power the LaunchPad by connecting the USB cable to a host PC. Then open a terminal window using TeraTerm or PuTTY on the appropriate COM port. To see your header message, press the Reset button on the LaunchPad to restart your application program.

Deliverables:

1. The video in *.mp4 format showcasing the Program that counts the number when SW3 is pushed and starts blinking LED when the SW2 is pressed. The video should also capture the messages printed via UART on the terminal.

2. The video should also capture you while explaining and demonstrating the working of the circuit. If you have made extra connections, please explain it to us through the video. For example, you can ask someone at home to film you or use the selfie camera to record yourself explaining the working and then use the back camera to capture the demonstration of the circuit. You can also create your own PowerPoint slides and walk us through it for explaining the circuit connection and your idea.

3. Try to limit the video to maximum of 5 minutes. The video and audio clarity should be decent enough for us to judge your work.

4. Zip the source files and video and upload it to the canvas, make sure the zip is not locked and the source files can be opened using text editor software. Please comment the code wherever necessary.

5. 25 points will be deducted for late submissions between 4-7 days, and submission that are late by 1 week will not be accepted/graded.